sides 'No. of cylinders' (in decimal) :Interleave value: (in decimal) @FREE Syntax: Free [devname] Usage : Displays number of free sectors on a device @GFX Syntax: RUN GF ce package for BASIC09 to do c K2 Syntax: RUN GFX2([path]<funct r BASIC09 to handle enhanced Syntax: none Usage : Graphics handle graphics/ windowing comm e : Give on-line help to users will st of help topics @IDENT Syntax: ter information from OS-9 memor emory -s = use single line output - ns at execution directory @INIZ Syntax: Iniz <devname>{,<devname>} Usage : Attach a device @INKEY Syntax: RUN INKEY([path],strvar) Usage : BASIC09 subroutine to input a s abort to the proc k to a memory nts of text files into memory a new directory mory module d Syntax: Merge <p output @MFREE mory @MODP dule in memory gs -c = compare e = link to module C = off, obvle, ntyle = change obvle at off(sd) to ntyle V = verify module M = mask IRQs U = unmask IRQs @MONTYPE Syntax: Montype [opt] Usage : Set m monitor m = monochrome m ge : Creates and links an OS OCS Syntax: Procs [e] Usage m Opts : e = display all pro e : Prints the current data d the current execution direct ew filename> Usage : Gives t unb <i-code module> Usage ntax: Setime [yy/mm/dd/hh:m ock @SETPR Syntax: Setpr <p ed process to num @SHELL Syntax: Shell <arglist> Usage : OS-9 command interpreter @TMODE Syntax: Tmode [.pathname] [params] Usage : Displays or changes the ope r /p> [value] nlink <modna ntax: Wcreate [op : Initialize and create windows Opts : -? = display help -z = read command lines from stdin -s=type = set screen type for a window on a new screen @XMODE

AUSTRALIAN OS9 NEWSLETTER

EDITOR	Gordon Bentzen	(07) 344-3881
SUB-EDITOR	Bob Devries	(07) 372-7816
TREASURER	Don Berrie	(07) 375-3236
LIBRARIAN	Jean-Pierre Jacquet	(07) 372-4675
SUPPORT	Brisbane OS9 Users Group	

Addresses for Correspondence

Editorial Material:

Gordon Bentzen
8 Odin Street
SUNNYBANK Qld 4109

Subscriptions & Library Requests:

Jean-Pierre Jacquet
27 Hampton Street
DURACK Qld 4077

Volume 4 December 1990 Number 11

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group
Volume 4 Number 11

EDITOR : Gordon Bentzen
SUBEDITOR : Bob Devries
TREASURER : Don Berrie
LIBRARIAN : Jean-Pierre Jacquet
SUPPORT : Brisbane OS9 Level 2 Users Group.

This edition of the newsletter will cover December and January as we like to have a break at this time of year, and this combined issue follows the pattern of previous years.

We are proud to announce that the National OS9 User Group is in receipt of an award for our efforts. The National OS9 User Group has been honoured by the "Coco-Link Award 1990". Those of you who also support the Coco-Link magazine produced by Robbie Dalzell as editor and Garry Holder as sub-editor would already be aware of this award, as a copy of it was included in the December edition of Coco-Link.

I must say that we were taken completely by surprise by this award and are most grateful for the recognition of our efforts to support OS9 and the Color Computer, and of course very pleased to receive the $100.00 cheque, payable to the National OS9 Usergroup, which is part of the Coco-Link award.

Our sincere thanks go to Robbie Dalzell and Garry Holder for this unexpected but very welcome recognition.
(It's always nice to receive a pat on the back.)

The COCO-LINK MAGAZINE is produced bi-monthly in South Australia and is well worth your support if you are interested in any aspects of the Tandy Color Computer. This magazine includes many interesting BASIC listings, hardware tips, general information, advertisements for hardware / software and even presents OS9 articles. If you do not currently subscribe then contact:-

Coco-Link Magazine
31 Nedland Cres.
Pt.Noarlunga Sth. S.A. 5167

Phone (08) 386 1647

SUBMISSIONS We are in urgent need of submissions for inclusion in this Australian OS9 Newsletter. Our more advanced members seem to be mostly interested in "C" programmes whilst new users are often requesting Basic09 listings and programmes. So how about it, every member must be able to submit something which is suitable for the public domain. Your submission does not have to be a masterpiece as we can all learn something from others at all skill levels. Maybe you have questions or a problem which someone in the usergroup will be able to solve.

The Australian OS9 Newsletter is intended to be a newsletter produced by the members of the National OS9 Usergroup. That is you. We simply collate all the information and present it here in the newsletter. The intent also is that this service is provided on a non-profit basis, hence no paid advertisements are included and no charges apply to members for any submissions, including private ads.

All submissions must be sent to us on DISK, preferably in OS9 format and as a "vanilla" ASCII file. We can however transfer "vanilla" ASCII files from RS-DOS and MS-DOS disks.

IN THIS EDITION Bob Devries presents some valuable information on transferring OS9 PROFILE files to SDF (System Data Files) and also presents the advantages of using the GFX2 module which is available from our P.D. library.

Don Berrie has included some insight for Multi-Vue users, and I have some comments about "SPANNER", a hard-disk backup and restore utility, also from the P.D. library.

Unfortunately our P.D. database is not yet completed so we are still not able to fill requests for a copy of it on disk. Work is continuing on this and it will be completed early in the new year.

We do hope that you have found something of interest in the newsletters presented during 1990 and that the article which you are going to submit will help other members enjoy their OS9 even more. Remember to look for our next newsletter edition in February 1991.

Cheers, Gordon.

MERRY CHRISTMAS TO ALL

Bob, Don, Jean-Pierre & Gordon.

We publish the following documentation, together with the C source code with the kind permission of the author Jerry Stratton, one of the more active contributors to the INTERNET OS9 mailing list. If you do not have the necessary C compiler to enable you to compile it, the compiled executable file is available from our public domain library. The usual rules for copying apply. (You will also need the Cgfx.l library replacement, together with the support header files in order to successfully compile it. This replacement library is also available in our PD library.)

OPTSTART

OptStart allows you to use the same script to do different things, without having to ask or wait. It takes one or two commands, and then executes one of the commands depending on whether a specified key is being held down.

For example, I have the following line in my startup script:

```
optstart 'rdisk </d0/Yes' 'rdisk r' </1
```

Optstart defaults to checking the SHIFT key. So, in this case, the command 'rdisk </d0/Yes' is executed if I am not holding down the shift key. This simply sets up my ramdisk. If I am holding down the shift key, it executes the command 'rdisk r'. This attempts to rebuild whatever ramdisk existed before I rebooted. (this is the RamDisk available from MicroCom).

You can use any of the following keys:

```
SHIFT
CTRL
ALT
LEFT    arrow
RIGHT   arrow
UP      arrow
DOWN    arrow
SPACE   bar
```

and these can be used in any combination. The following command:

```
OptStart 'echo up' 'echo down' -shift -ctrl
```

will echo the word 'down' if both the shift and control keys are held down. Otherwise, it will echo 'up'.

```
OptStart 'mega' -shift -ctrl -alt </1
```

in your startup script will execute the 'mega' command (setting up 1 Meg) UNLESS you are holding down the SHIFT key, the CTRL key, and the ALT key.

Note that the </1 is required in scripts, which is where you'll probably be using OptStart most often.

Jerry Stratton
CRIT@usdcsv.acusd.edu

----------CUT HERE----------

```
/* OptStart [IfUp] IfDown [-optkey] (c) 1990 Jerry Stratton */

#include <stdio.h>
#include <keysense.h>

#define StdIn 0
```

```
#define StdOut 1
#define StdErr 2

main (argc,argv)
int   argc;
char  *argv[ ];

{
   int   Keys=0;
   char *Down="\0";
   char *Up="\0";

   /* exit quietly if there are no arguments */
   if (argc<2)
      exit(0);

   /* check for options and commands */
   while (--argc)  {
      if (*argv[argc]=='-')
         Keys+=GetKey(argv[argc]+1);
      else if (*Down)
         Up=argv[argc];
      else
         Down=argv[argc];
   }
   /* if no conditions were given, exit quietly */
   if (*Down==0)
      exit(0);

   /* if no options were given, assume shiftkey */
   if (Keys==0)
      Keys=SHIFTBIT;

   /* check if the Key(s) are down */
   if ((Keys&_gs_ksns(StdIn))==Keys) {
      system (Down);
      wait();
   }   else  if (*Up) { /* if an Up branch exists, do it */
      system (Up);
      wait();
   }
}

/* find out what key the option is */
GetKey (Option)
char *Option;
{
   int   Count=0;
   static struct {
       char *OptName;
       int  Mask;
       } Opts[ ]= {{"SHIFT",SHIFTBIT},
                   {"CTRL",CTRLBIT},
                   {"ALT",ALTBIT},
                   {"UP",UPBIT},
                   {"DOWN",DOWNBIT},
                   {"LEFT",LEFTBIT},
```

```
                {"RIGHT",RIGHTBIT},
                {"SPACE",SPACEBIT},
                {"dummy",0}};

   while (Count<sizeof(Opts) && strucmp(Opts[Count].OptName,Option))
      ++Count;
   return(Opts[Count].Mask);
}
```

OS9 Profile to SDF conversion.

by Rob Devries.

Recently there has appeared a new OS9 database programme, namely Data-Windows, written by Keith Alphonso and sold by Alpha Software Technologies. The idea behind this database programme is quite good, in that is uses OS9's windowing capabilities to its fullest extent. Multiple windows may be set up to simultaneously display data and reports in different ways, and with different key fields.

The programme is not without its problems and minor bugs, however. One of these buglets is that it, like one of Mr. Alphonso's other programmes, 'DMTree', it does not leave the window the way it found it when it quits.

Another rather annoying problem is that the manual is written as if the user is expected to be an expert in both the use of databases, and indeed the programme itself! Quite a few features are merely skimmed over, and others are not mentioned at all. A mention is made of a file format called DB9-90, but nowhere is this file format explained. To find this information, you will be required to purchase extra software from the company.

Without going into the trials and tribulations I went to to try and get a working database using the copy lent to me by one of our members for review purposes, I will get to the point of this article. I wanted to import a database from OS9 Profile, and, according to the Data-Windows manual, it is capable of importing data in SDF format, which is a file format that quite a few other database programmes use for exporting files. Not so OS9 Profile, however. It only outputs data in DynaCalc spreadsheet format. That's fine if you want to put data into your spreadsheets, but not much use for anything else. Now what my little C programme does is read the Dynacalc file format that Profile writes, and outputs an SDF format file. For those who are not familiar with SDF files, here's what they look like:-

"field 1","field 2",.....,"field n"<CR>
this is record 1
"field 1","field 2",.....,"field n"<CR>
this is record 2
.
.
.

And so on for all the records in the database file.

So you see that fields within records are separated by commas, and surrounded by inverted commas, and records are separated by carraige returns. By the way, here I encountered what to me is a rather serious bug in OS9 Profile, namely that you cannot create a spreadsheet file from a database that has anything other than text data fields (type 1). If you do, Profile crashes! Usage of the conversion programme, after you've created the spreadsheet file with Profile, is simple. Here it is:-

Convprofile <dyna_file >sdf_file

Note that both input and output redirection is used. So without further ado, here's the code for my little utility.

```
/* Conversion programme to change OS9 Profile spreadsheet file */
/* to SDF file. Accepts STDIN for input, and writes STDOUT.    */
#define BUFMAX 1000

main()
{
    char *inbuffer;     /* pointer to memory returned by malloc() */
    int nc = 1;         /* integer counter preset to 1 for while loop */
    int flag = 0;       /* end of record flag */
    char *malloc();     /* define function as returning char pointer */
```

```
    inbuffer = malloc(BUFMAX);    /* grab a chunk of memory to store file lines */
    while (nc > 0)     /* keep reading until nc = 0 (end of file reached) */
        {
        nc = readln(0,inbuffer,BUFMAX);    /* grab a line of text */
        if((strncmp("/row",inbuffer,4) == 0) || (strncmp("/ROW",inbuffer,4) == 0))
            continue;  /* ignore special codes for DynaCalc */
        if(inbuffer[0] == '*')   /* and comment lines */
            continue;
        if((inbuffer[0] == '@') && (inbuffer[1] == '\n'))
            {
            write(1,"\n",1);   /* if @ is found, end of record is reached */
            flag = 0;           /* so print a carraige return */
            continue;
        } else {
            if (flag)
                {
                write(1,",",1); /* if not first field, write a comma */
            } else {
                flag = 1;       /* else it was first field, so toggle flag */
            }
        }
        write(1,"\"",1);       /* write a field separator */
        if(inbuffer[0] == '\'')    /* strip out apostrophe chars inserted by Profile */
            writeln(1,inbuffer+1,nc-2);    /* write the line */
        else
            writeln(1,inbuffer,nc-1);
        write(1,"\"",1);                /* write a filed separator */
    }                                    /* and go around the loop  */
    free(inbuffer);                      /* free up memory          */
}
```

oooooooooo0000000000oooooooooo

HARD DISK BACKUPS

We have in past editions commented on the wisdom of making regular backups of a hard disk. To quote one of our members (D.B.)"there are two types of hard disk users; those who have just done a backup, and those who are about to have a crash!"

Now one can understand why we probably all get a little slack when it comes to making a backup of the hard drive connected to the CoCo because it can often turn into a major project which can take several hours. This article is based on some personal experience with some of the available hard disk utilities and while I do not claim to be an expert on the subject, I hope that you find it useful.

I run a 20 meg hard drive using the Burke & Burke interface and standard P.C. type controller. Burke & Burke supply hard disk utilities with their XT interface which include "hdb" & "hdr" (hard disk backup & hard disk restore). Also available separately is "fsr", a hard disk file system repack utility which reads and rewrites files to eliminate fragmentation of the hard disk. These utilities work just fine, however the process can take many hours.

HDKIT is another set of backup and restore utilities by Pete Lyall which is available from the public domain library. The description of the utilities directly from the hdkit.doc file reads, "The suite of backup tools in this archive is a collection of programmes that are written, maintained, documented and updated by Pete Lyall. Some of them are based on early versions of similar tools written by Dan Connolly and those provided for a similar purpose in the Unix system (i.e. CPIO). This toolkit is SHAREWARE/CHEAPWARE. Please read the last paragraph of this file before going on." (end quote)

The tools include, Files, Archdir, Bigfile & Restore. Hdkit uses pipes which tend to slow down its operation somewhat.

Example :-

```
files   -bedge-30    !   archive   /dd   -vl
```

provided for a similar purpose in the Unix system (i.e. CPIO). This toolkit is SHAREWARE/CHEAPWARE. Please read the last paragraph of this file before going on." (end quote)

The tools include, Files, Archdir, Bigfile & Restore. Hdkit uses pipes which tend to slow down its operation somewhat.
Example :-
files -bedge-30 ! archive /dd -vl /dd/log/monthly.log #40k ! bigfile /d0/part0 /d1/part1 /d2/part2 #40k

The Hdkit suite works just fine, in fact I have used it for several backups and a couple of restores.

The most difficult part of using Hdkit is remembering the correct syntax of command modifiers and command lines.

REBACK is a basic09 procedure written by Jeff Blower which is a front end for Hdkit which makes it menu driven and remembers the command syntax for you. This procedure I have found to be very useful for backing up changed files or files created since the last backup. It saves a file "backup.date" in the /dd/sys directory so that you don't even have to remember the date of the last backup.

SPANNER This is a utility written in C by Jay Heisse which also uses some of the tools from HDKIT.

SPANNER copies files (named on stdin) to removable media,
-or- restores files/directories if -r option given

options: -r w -l[=logfilename] /dev1 /dev2 /dev3 /dev4

-r restores files from floppies to CURRENT directory
-w writes files to floppies, (default)
-l logfile, logs messages to /R0/blog by default.
-l=FILENAME to change from default.

/dev default backup drive is /F1, but a drive or drives may be given.
In the case of multiple drives, they will be used in sequence before prompting for more media.
Drivename arguments are recognized by the leading slash.

example:

chd /h0
files -ebdg-8!spanner /f0 /f1 -l=/dd/log/friday.bkup

For info on permanently changing defaults and other details, use 'man spanner'

Personally, I find it most convenient to use "files" to create a backup.list which can be edited prior to running the backup. Here are the command lines and procedure which I use.

files -etdgYY/MM/DD >/r0/backup.list

What this does is;
E = expand to all directories
T = total files and estimate disks needed for backup
DG = dates of files greater than yy/mm/dd

Now, edit the file "backup.list" to delete any files which are not to be included in the backup. OS9Boot & ALTBoot are a couple which I normally delete as I use "bootport" to restore the kernels to the correct location on the disk along with OS9Boot etc.

Now to commence the backup;

spanner /d0 /d1 -L=/r0/blog #32k </r0/backup.list

In this example, spanner will use floppy drive /d0 for "part1" /d1 for "part2" and prompt for more media once the first two disks are filled. A log of all files archived will be created on /r0 called "blog" (backup log). Spanner will read pathnames and files from "backup.list" and include only the files listed in this file.

The best part of all is the time saving. My hard drive has about 24,000 sectors in use by files, and Spanner does a full backup of all files in under one (1) hour, including floppy formatting. Spanner has a toggle which can be changed between floppies to turn ON & OFF the floppy format function.

To restore the files from the floppy backup set, just add the -r option (restore)

spanner -r /d0 /d1 -L=/r0/blog #32k

The above examples assume that device descriptor /dd refers to the hard drive, which will normally be the case.

Gordon Bentzen.

oooooooooo0000000000ooooooooo

The New GFX2

Those of you who read the magazine 'The Rainbow', would no doubt have seen the references to a 'new' GFX2 module for use with Basic09. Indeed, many were the curses heard by me when Basic09 programmes, listed in that magazine, could not be used because a 'new version' was required. Well, now you can all use the new GFX2 module, because it is available from the software library. The copy we have in the library is a sort-of pre-release version, but all the necessary parts are there. The programme archive is available in the usual way from our software librarian, Jean-Pierre Jacquet.

I have taken one of the documentation files from the archive, and reproduced it here for you to whet your appetites. have a read of this:-

WINDOW COMMANDS
==========

Previously, setting up menued windows, or even getting the mouse packet, involved using the Syscall subroutine and information from the developer's documentation. I believe this has stymied the casual programmers a lot. In an attempt to make things easier, GFX2 now supports the more popular windowing stat calls. (See "gfx2.man")

There are four new calls used to set up a menued window: TITLE, MENU, ITEM and WNSET. They require some program storage for the main window descriptor and each of the menus, which is most easily set up by defining string arrays. The main window descriptor variable must have a dimension of at least 2; otherwise the dimension should be at least the same count as the number of menu bar selections (across the top: like FILES, TANDY, etc), plus 1 more.

Each menu bar selection may have items that are in its pulldown menu. Each pulldown requires an array dimensioned to at least the number of items.

SETTING UP MENUS
==========

The new commands are much easier to use than to explain. In addition to the examples on disk and the Details here, let us set up a small menued window program as a quick example...

Let's say that you wanted to write a program called "Tools", that had two menu selections (in addition to the close box, which is automatically defined). These menus are called "Disk" and "Memory". Under Disk we want "Dir", "Free", "PWD", and "Format". Under Memory we want "MFree" and "Procs".

First, we must set aside storage for OS-9 and GFX2. Since we have two menubar selections, we define a window descriptor array of size 2+1 = 3.

```
DIM wd(3):STRING \ (* This is the window descriptor)
```

We also need to define storage for each pulldown menu. Our first selection has three items, our second has two. (NOTE: if the number of items might change, simple give it more size accordingly).

```
DIM m1(3):STRING \ (* Disk selection
DIM m2(2):STRING \ (* Memory selection
```

We need to assign an ID number to both menu selections. Numbers 1-32 are reserved for the system, so let's use 33 and 34.

```
DIM MId_Disk,MId_Mem:INTEGER
MId_Disk=33
MId_Mem =34
```

We can also define some numbers that will make reading the program a little easier:

```
DIM Disable,Enable:INTEGER
Disable=0
Enable=1
```

The rest is easy! Simply fill in the blanks for the following commands: Set the title name, minimum window size, number of menu selections:

```
RUN gfx2("Title",wd,"Tools",34,10,2)
```

Set up each menu selection position,name,ID,width,itemcount,items,enable:

```
RUN gfx2("Menu",wd,1,"Disk",MId_Disk,8,4,m1,Enable)
RUN gfx2("Item",m1,1,"Dir   ",Enable)
RUN gfx2("Item",m1,2,"Free  ",Enable)
RUN gfx2("Item",m1,3,"PWD   ",Enable)
RUN gfx2("Item",m1,4,"Format",Disable)

RUN gfx2("Menu",wd,2,"Memory",MId_Mem,5,2,m2,Enable)
RUN gfx2("Item",m2,1,"MFree",Enable)
RUN gfx2("Item",m2,2,"Procs",Enable)
```

NOTE: the order isn't important above. It's just easier to read this way. Now that we have defined the window, selections, and pulldowns, we can tell OS9 to set up a framed (type=1) window:

```
RUN gfx2("WnSet",1,wd)
```

You can disable or change the menus on the fly.

USING THE MOUSE

Now that the menued window is set up, there are several utility calls that are very helpful. First, we need to turn on the gfx cursor and also the autofollow (since we'd rather not keep telling OS9 where to put the cursor!).

```
RUN gfx2("SetMouse",3,1,1) \ (* Turn on autofollow
RUN gfx2("GCSet",$CA,1)    \ (* Use standard arrow pointer
```

The Mouse call will tell us the status of the pointing device:

```
DIM valid,fire,mx,my,area,sx,sy:INTEGER \ (* Place at program front!
RUN gfx2("Mouse",valid,fire,mx,my,area,sx,sy)
```

We could simple loop around, checking the mouse and the keyboard (INKEY), but quite often a program need only wait for the user to click his fire button, and only then do we go after the menu selection. This is much more friendly to other programs running, and aids in multitasking quite a bit:

```
RUN gfx2("OnMouse",0)  \ (* Sleep until mouse clicked
```

This will return only when this window is the interactive one, and the user has clicked his mouse button. Then we are woken up, and can figure out what he meant us to do:

GETTING MENU SELECTIONS

On wakeup, we need to see if the user is really clicking on the menu area.

```
RUN gfx2("Mouse",valid,fire,mx,my,area,sx,sy)
```

If valid isn't zero, and the right button is down, and the area is the menu area, then we know what the user meant. So we can call the Get Selection routine to let the user pulldown a menu or return a bar selection like Close.

```
DIM menu_id, menu_item:INTEGER \ (* This should be at front of program!
RUN gfx2("GetSel",menu_id,menu_item
```

The menu_id will be the menubar selection (00 if none), and the menu_item will be the specific pulldown item number.

The total loop to wait for a click, see if menu, get the selection, is:

```
LOOP
  RUN gfx2("OnMouse",0)
  RUN gfx2("Mouse",valid,fire,mx,my,area,sx,sy)
  IF valid<>0 AND fire=1 AND area=1 THEN
   RUN gfx2("GetSel",menu_id,menu_item)
     IF menu_id <> 0 THEN
      GOSUB xxx \ (* go do selection number *)
     ENDIF
  ENDIF
ENDLOOP
```

You could also wait for the mouse click in a drawing program, for example, and if it wasn't on the menubar area (area=1) then you could start a drawing loop or whatever.

For those of you who write programmes in Basic09, this will really take the work out of using the Multi-Vue windowing manager, with its pulldowns and mouse selection. I expect to see a flood of requests for usergroup software after this.

Rob Devries.

oooooooooo0000000000oooooooooo

STARTING MULTIVUE

Most, if not all of CoCo OS9 Level 2 users occasionally use the Multi-Vue desktop graphical interface on their machines. With the patches which are available for gshell, it makes a reasonably good GUI (Graphical User Interface) which runs at an acceptable speed. All of the patches for speed and operational improvement are available from our Public Domain Library.

One of the things that has irked me for some time, was the fact that multistart really does not leave your system in the state in which it existed, before Multi-Vue was run. The main problem involves the replacement on an existing window with a graphics window. If you are like me, you only use graphics windows when you absolutely have to, and to have one of your existing windows changed to a hi-res graphics window, often with proportionally spaced hi-res fonts in operation, is not acceptable.

To get around this problem, I wrote the following utility to replace MultiStart. Source code for both C and Basic09 versions are included for you. The utility is simply named mvu, and uses the system default settings to start multiview. (ie Mouse Resolution and Port, as well as colours initially selected.) Changes to these settings could be fairly simply added to the code, however I will leave that to you to do. Something to keep you occupied over the Christmas holidays. (grin).

The only other thing to look out for (see comments in sources) is the setting of the execution directory. You will have to change this to reflect your own system setup. In other words,

the location of your MultiVue commands needs to be
set in that particular line, or the programme will not work.

Cheers ... Don Berrie.

C Source Code

```
/* Mvu.c -- simple Multistart replacement -- (c) 1990 D.Berrie */

#include "/h0/usr/ccomp/defs/stdio.h"
#include "/h0/usr/ccomp/defs/os9.h"

int wpath;
struct registers regs;

main()
    {
    char devnam[32];
    char systr[50];
    int *mvprno;
    systr[0] = '\0';
    wpath=open("/w",3);
    DWSet(wpath,7,0,0,80,24,0,0,0);
    getdev(devnam,wpath);
    chxdir("/h0/cmds/xcmds"); /***** Change for your own system *****/
    Select(wpath);
    strcpy(systr,"gshell <>>>");
    strcat(systr,devnam);
    system(systr);
    }

getdev(outstr,pathno)
char outstr[32];
int pathno;
    {
    char temstr[32];
    char *ptr;
    int i = 0;
    outstr[0] = '/';
    outstr[1] = '\0';
    ptr = temstr;
    regs.rg_a = pathno;
    regs.rg_b = 0x0e;
    regs.rg_x = ptr;
    _os9(I_GETSTT,&regs);
    for (i=0; temstr[i] < 0x7d && temstr[i] > 0;i++);
    temstr[i] = temstr[i] & 0x7f;
    temstr[i+1] = '\0';
    strcat(outstr,temstr);
    }
```

BASIC09 Source Code

```
PROCEDURE mvu

(* Mvu -- Simple Multistart replacement -- (c) 1990 D.Berrie *)
```

```
DIM devnam:STRING[32]
DIM systr:STRING[50]
DIM wpath:BYTE
OPEN #wpath,"/w":UPDATE
RUN gfx2(wpath,"dwset",7,0,0,80,24,0,0,0)
RUN getdev(devnam,wpath)
CHX "/h0/cmds/xcmds" (***** Change for your own system *****)
RUN gfx2(wpath,"select")
SHELL "gshell <>>>"+devnam

PROCEDURE getdev
ON ERROR GOTO 1000
TYPE registers=cc,a,b,dp:BYTE; x,y,u:INTEGER
DIM regs:registers
PARAM devnam:STRING[32]
PARAM wpath:BYTE
DIM i:INTEGER
DIM callcode:BYTE
DIM errnum:BYTE
regs.a=wpath
regs.b=$0E
regs.x=ADDR(devnam)
callcode=$8D
RUN syscall(callcode,regs)
FOR i=1 TO 32
EXITIF MID$(devnam,i,1)>CHR$(128) THEN
devnam="/"+LEFT$(devnam,i-1)+CHR$(ASC(MID$(devnam,i,1))-128)
ENDEXIT
NEXT i
END
1000 RUN closerr(wpath,errnum)
END
```

oooooooooo0000000000oooooooooo

DATAMODULES

Do you have a floppy disk system? How many times have you typed the line:

```
chd /d0;chx /d0/cmds
```

If you are like me, that is about the only line that you can touch-type!! Before I purchased my hard disk, I suppose I typed just that line maybe 20 or 30 time each session!! You know, everytime that you wanted to run something different, and you did not have the correct disk in your (single?) drive,you needed to type this string.

OK so now you've spent your dollars, and puchased a hard drive. Now the problem becomes:

```
chd /h0/usr/ved/docs;chx /h0/usr/ved/cmds
```

This is really no better!!

Well an answer is at hand. Using packed data modules, we can simply replace these strings with a (perhaps) two or maybe three letter command. Here is the process.

First of all build your string, using a text editor or whatever, maybe you could even use edit. Just a simple example at first.

EDIT newd (for newdisk)

```
 chd /d0;chx /d0/cmds
 command
q(uit)
```

Then, using datamod (available from our PD library) pack the module.

```
DATAMOD nd <newd >nd

ATTR nd e pe
```

LOAD nd

Now when you type nd, the command string "chd /d0;chx /d0/cmds" and then the executable file will be executed!!! Neat hey?

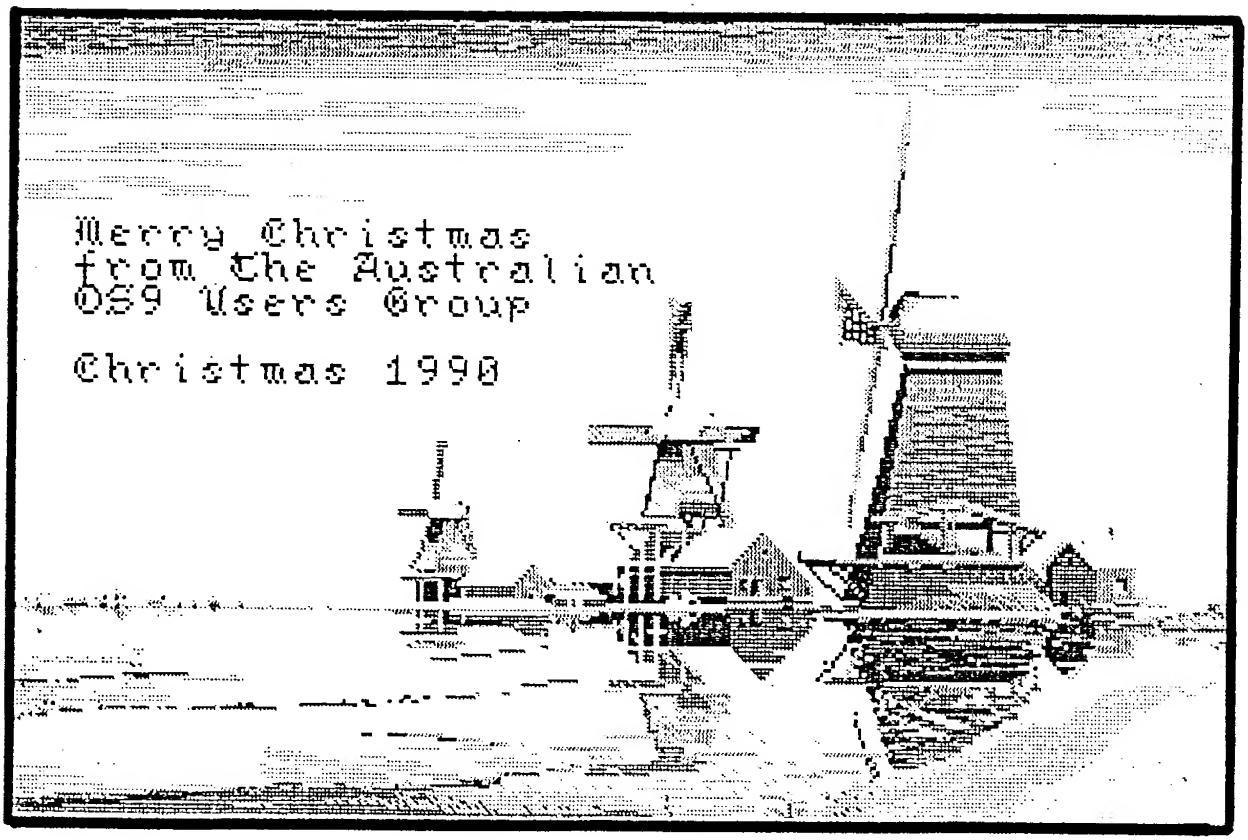
Another variation would be to use exactly the same string, but enclose it in brackets. That is, "(chd /d0;chx /d0/cmds)". Go through the same process again, this time giving it a different name, say nsd, and not using a particular executable file to run. Then when you load and execute the command, "nsd", the system will start a new shell, in the current window, and proceed to change the directories. Then you can rau any process from that shell that you wish. When you have finished with that particular disk, simply type <CONTROL> + <BREAK> and replace the disk. You will then be located back where you came from.

As you can see, the possibilities are endless!!!

One further enhancement, each of these packed data modules are extremely small (in terms of the number of bytes of memory used), and so a large number of them can be crammed into one 8K block by merging them together before they are loaded. You may even have enough space left with your modules merged with SHELL to fit in a few. That way, you get all of the advantages, and maybe use NO memory!!

Happy Christmas ... Don Berrie.

oooooooooo0000000000ooooooooo



Coco-Link Award

This is to certify that

The National OS9 User Group

is commended for meritorious service to the Color Computer Community of Australia

PRESENTED BY COCO-LINK MAGAZINE

December 1990

Robert Dalzell

EDITORS

The COCO-LINK AWARD is presented annually to individuals or organisations who the editors of COCO-LINK consider have contributed something of value to the Coco Community in Australia. $100.00 is awarded each year.

This year's COCO-LINK AWARD goes to the NATIONAL OS9 USER GROUP for organising and maintaining this very important section of the Australian Coco world.

I think we all owe a debt of gratitude to Gordon Bentzen, Bob Devries and Don Berrie for taking up the challenge and bringing new life to the OS9 User Group after it had fallen by the wayside. These three gentlemen deserve all the accolades we can bestow on them.

The monthly newsletter of the National OS9 User group carries information and programming for the budding, as well as the more advanced OS9 user. The Group also maintains a large library of OS9 Public Domain software which is available to subscribers at minimal cost. The three office holders named above have always been found willing to help straighten out the problems of the less informed.

In presenting the $100.00 cheque to the NATIONAL OS9 USER GROUP we hope, in some small way, to help ease the financial burdon of running a national group such as this and, in so doing, insure the continuance of the group.

We heartily congratulate you on a job well done.

Robbie Dalzell & Garry Holder
For COCO-LINK Magazine.

(Details on how to subscribe to the NATIONAL OS9 USER GROUP can be found in the Noticeboard page in this magazine).